School of Computation, Information and Technology
Technische Universität München

Lehrstuhl für
Rechnerarchitektur & Parallele Systeme
Prof. Dr. Martin Schulz
Dominic Prinz
Jakob Schäffeler

Lehrstuhl für
Design Automation
Prof. Dr.-Ing. Robert Wille
Stefan Engels

Einführung in die Rechnerarchitektur

Wintersemester 2024/2025

Übungsblatt 10: Parallelisierung und Caches 23.12.2024 – 10.01.2025

1 Speedup

Wie viel Speedup kann man theoretisch bei den folgenden C-Programmen durch Parallelisierung bei Nutzung von 4 bzw. 16 CPU-Kernen erreichen?

a) Verwenden Sie für die Berechnung des Speedups das Amdahlsche Gesetz: $S_{\text{Amdahl}} = \frac{T}{t_s + \frac{t_p}{n}}$

Halten Sie sich dabei an die folgenden Teilschritte:

- Ist das Programm parallelisierbar?
- Welche Zeilen können parallelisiert ausgeführt werden? Von wie vielen Threads maximal?
- Berechnen Sie jeweils die gesamte Laufzeit für parallelisierbare und seriell ausgeführte Programmteile.
- Nutzen Sie diese Werte zur Berechnung des Speedups.

Hinweis: Sie können von der (idealisierten) Annahme ausgehen, dass alle Zeilen, auch Schleifenstrukturen wie for(...) und while(...) bei jedem Schleifendurchlauf gleich viel Zeit benötigen. Die benötigte Zeit (für jede Iteration im Falle einer Schleife) ist hinter jeder Zeile angegeben.

i) Das Zählen der Elemente in einer verketteten Liste.

```
1 struct item {
2     int data;
3     struct item* next;
4 };
5
6 int getCount(struct item* head)
```

```
7  {
8      int count = 0;
9      struct item* cur = head;
10     while (cur) {
11         ++count;
12         cur = cur->next;
13     }
14     return count;
15 }
```

1) Nicht parallelisierbar da keine feste Problemgröße (verkettete Liste)

ii) Den Elementen eines Arrays einen Wert hinzufügen

```
1  float x[100];
2  ...
3  void add_to_x(float base, float exponent) {
4      float value = pow(base, exponent);     //100ns
5      for(int i = 0; i < 100; i++) {         //3ns
6          x[i] += value                      //17ns
7      }
8  }
```

1) ja, weil Problemgröße fest
2) Bedingung 101 mal geprüft → mehr als 101 Threads

3) $t_p = (17ns + 3ns) \cdot 100 + 3ns = 20 \cdot 100ns + 3ns = 2003ns \approx 2{,}0\,\mu s$
$t_s = 100ns = 0{,}1\,\mu s$

4) Aufteilzeit mit 1 Prozessor $T = t_p + t_s = 2{,}1\,\mu s$
Speedup $(n_{p=4}) = \frac{T}{t_s + \frac{t_p}{n}} = \frac{2100ns}{100ns + \frac{2000ns}{4}} = 3{,}5$
Speedup $(n_{p=16}) = \frac{2100ns}{100ns + \frac{2000ns}{16}} = 9{,}33$

b) Das Gustafsonsche Gesetz ist eine Erweiterung des Amdahlschen Gesetzes. Allerdings wird im Unterschied dazu ein festes Zeitfenster betrachtet, und die Problemgröße wächst proportional zur Anzahl der verfügbaren Prozessoren.

Für die Berechnung des Speedups nach Gustafson betrachten wir ein Problem mit n-fachem parallelem Teil als Basis. Dieses kann auf einem Prozessor in $t_s + n \cdot t_p$ berechnet werden.
Auf n Prozessoren kann der parallel ausführbare Teil gleichmäßig auf alle Prozessoren aufgeteilt werden. Somit erhalten wir als Ausführungszeit für n Prozessoren

$$t_n = t_s + \frac{n \cdot t_p}{n} = t_s + t_p$$

.

Der Speedup, der auf n Prozessoren erreicht werden kann ist somit beschrieben durch:

$$S_{\text{Gustafson}}(n) = \frac{t_1}{t_p} = \frac{t_s + n \cdot t_p}{t_s + t_p} = \frac{t_s + n \cdot t_p}{T}$$

Oder mit $t_s + t_p$ normalisiert mit 1:

$$S_{\text{Gustafson}}(n) = \frac{t_s + n \cdot t_p}{1}$$

Berechnen Sie den Speedup nach Gustafson für die Programmteile aus Aufgabe a)

Was hat der Unterschied zu bedeuten?

2 Roofline Modell

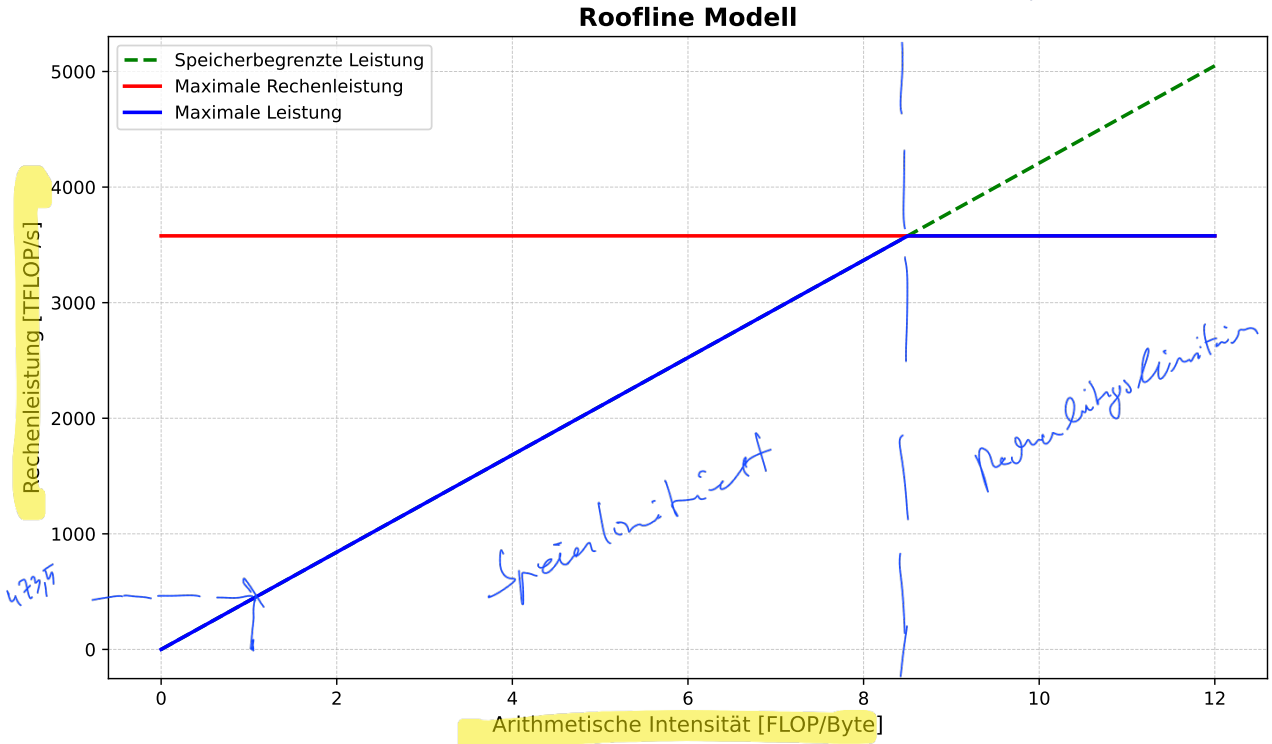


Abbildung 1: Roofline Modell

Verwenden Sie das in Abbildung 1 gegebene Roofline Model, um die maximale Performance für die untenstehende Programme zu berechnen. Gehen Sie davon aus, dass alle Fließkommazahlen 32-Bit groß sind. Halten Sie sich dabei an folgendes Vorgehen:

- Bestimmen Sie die arithmetische Intensität I der wiederholten Durchführung der beschriebenen Operationen in FLOP/Byte. Bestimmen Sie hierfür zuerst wie viele Floatingpoint Operationen ausgeführt werden und wie viele Bytes gelesen und geschrieben werden. Benutzen Sie dann zur Berechnung von I folgende Formel:

$$I = \frac{W}{Q} = \frac{\text{Anzahl der Operationen in FLOP}}{\text{Benötigte Speichertransfers in Bytes}}$$

- Verwenden Sie Ihre Ergebnisse um aus dem Roofline-Diagramm die theoretisch maximale Leistung für die Operationen in FLOP/s abzulesen.
- Sind die Probleme rechen- oder bandbreitenlimitiert?

Programme

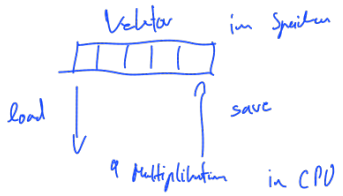
a) `String Copy`: Kopieren eines langen Strings aus dem Speicher an eine andere Speicherstelle.

b) `Potenzieren von Zahlen`: Laden eines Vektors aus Fließkommazahlen aus dem Speicher, 10-fache Fließkomma-Multiplikation des Zahlvektors mit sich selbst (also insgesamt 9 Multiplikationen), abspeichern an der ursprünglichen Speicherstelle.

32 Bit FL Zahl

a) $I = 0$ Nur Speichertransfers, keine FLOP

b)



$$I = \frac{W}{Q} = \frac{9}{2 \cdot 4\,\text{Byte}} = 1{,}125\ \frac{\text{FLOP}}{\text{Byte}}$$ Laut Tabelle → max 473,5 TFLOP/s

Speicherbandbreite limitiert

3 Auswertung von Speedupdiagrammen

Sie organisieren in einem Rechenzentrum die Ressourcenverteilung und bekommen untenstehendes Speedupdiagramm für fünf Programme vorgelegt. Beobachten Sie die Kurve und diskutieren Sie die Abwägungen bei der Auswahl der Anzahl der Prozessoren für jedes Programm. Beobachten Sie den maximalen Speedup und begründen Sie dessen Ursprung.

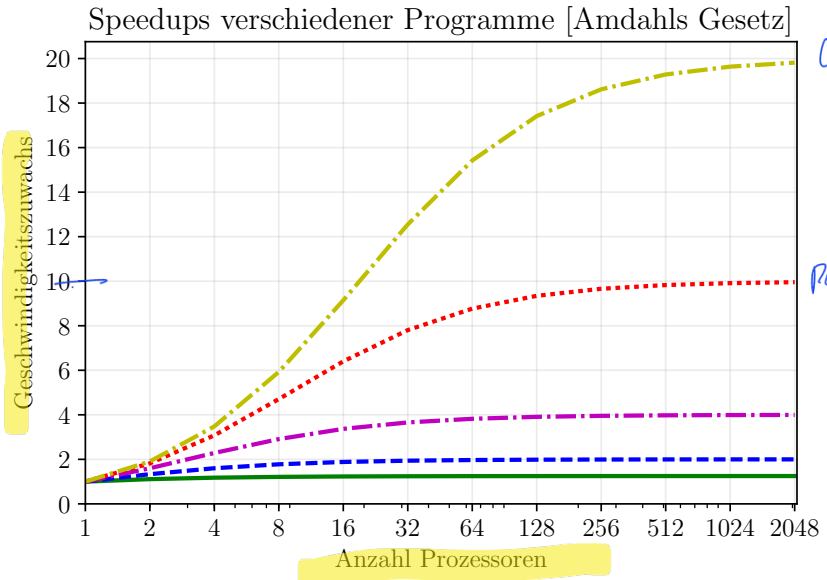


Abbildung 2: Speedup nach Amdahls Gesetz fünf verschiedener Programme

a) Wie groß ist der parallelisierbare Anteil der fünf Programme?

b) Das in Abbildung 3 dargestellte Diagramm zeigt den berechneten Speedup eines Programms und den real gemessenen. Wie kommt der sichtbare Unterschied zustande?

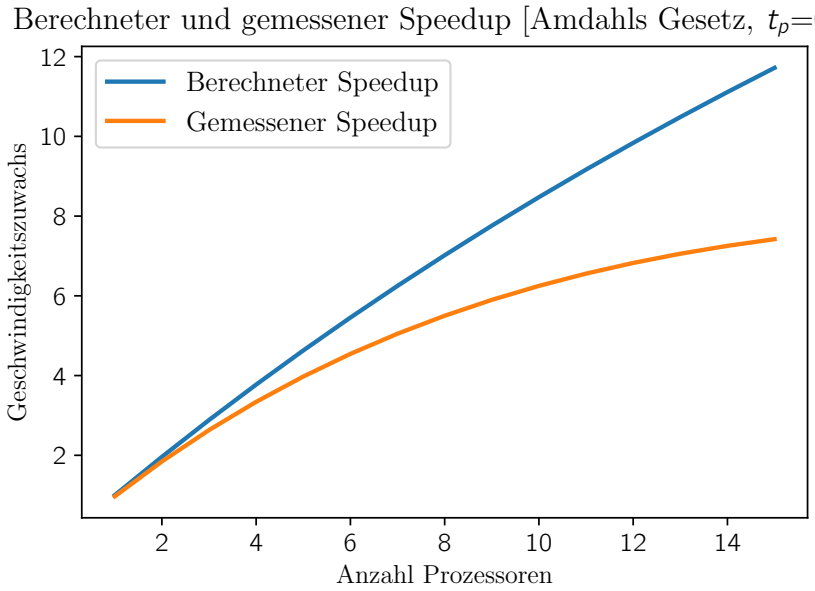


Abbildung 3: Unterschied gemessener und berechneter Speedup, Beispiel 1

c) Das in Abbildung 4 dargestellte Diagramm zeigt den berechneten Speedup für ein anderes Programm für große Datenmengen und den real gemessenen. Wie kommt der sichtbare Unterschied zustande?

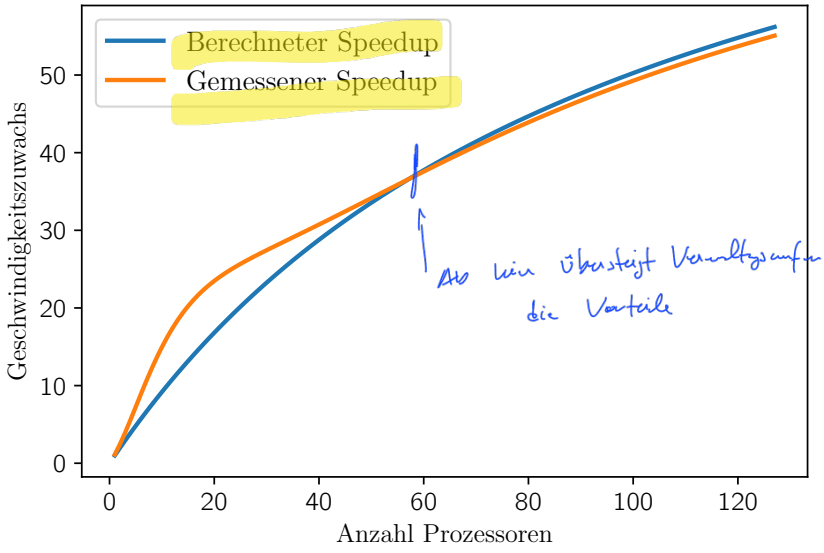


Abbildung 4: Unterschied gemessener und berechneter Speedup, Beispiel 2

d) Was kann man generell anhand eines Speedupdiagramm über die Rechenleistung eines Programms aussagen? Was nicht?

Speedup sagt aus wieviel schneller ein Code ist, wenn er parallelisiert wird

keine Aussage über Dauer / Alg. Komplexität / Effizienz

4 Hausaufgabe

Bearbeitung und Abgabe der Hausaufgabe 10 auf `https://artemis.in.tum.de/courses/401` bis **Sonntag, den 12.01.2025, 23:59 Uhr**.